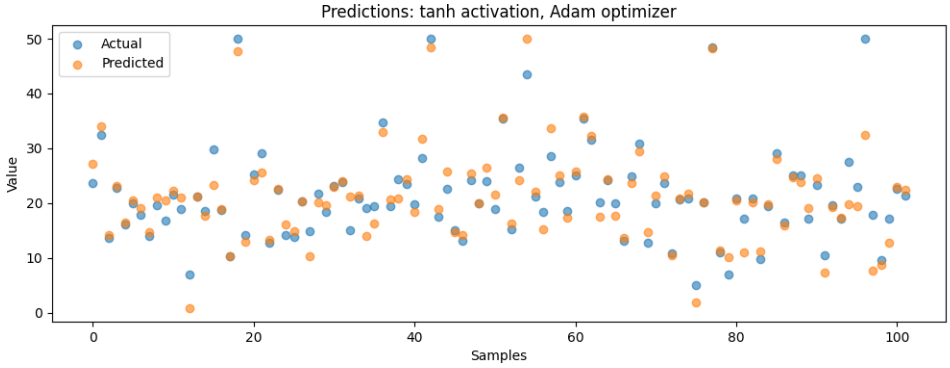
HW 6: Neural Networks Part 2 Report

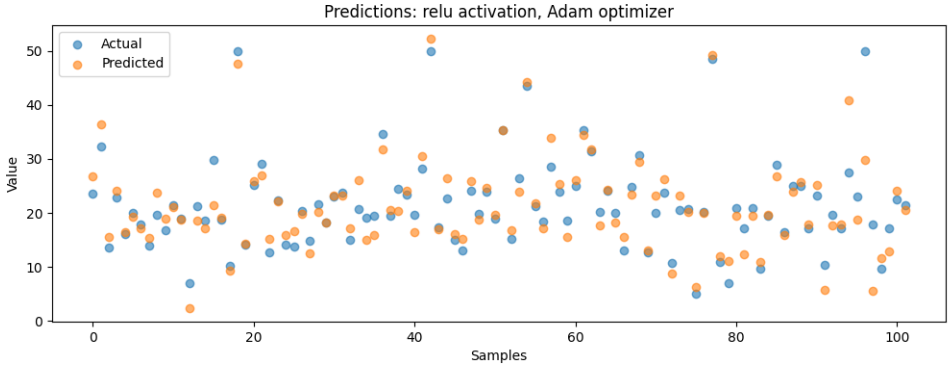
Keras vs.PyTorch

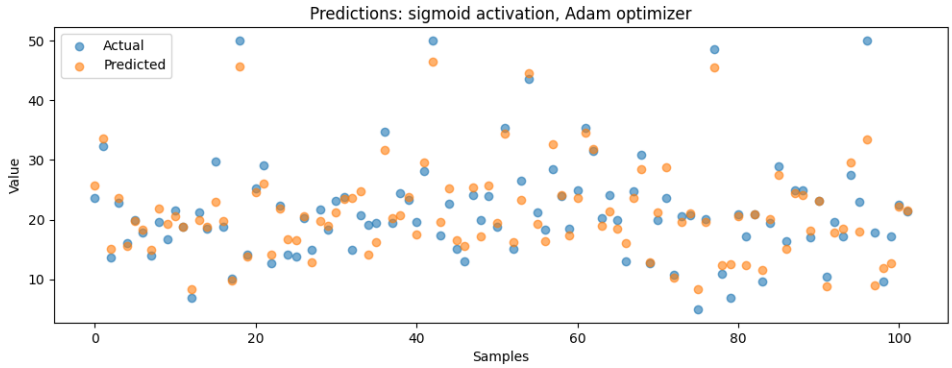
In terms of predictions on the testing data, the performance difference between the Keras and PyTorch models is negligible for both classification and regression. However, while the predictive performance of the models is comparable, the training time for the PyTorch models is considerably faster. This performance gain comes at the cost of a significant increase in implementation complexity.

Differences In Activation Functions

Changing the activation function had no significant impact on the prediction capabilities of the model. As shown in the graphs below, there is some variance in the predictions for individual data points between the different functions; however, the average accuracy for the regression models remains similar. This is also replicated in the classification models.





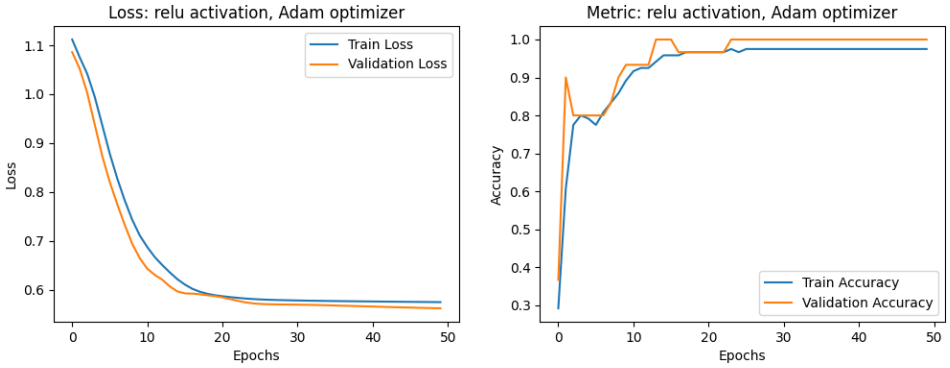


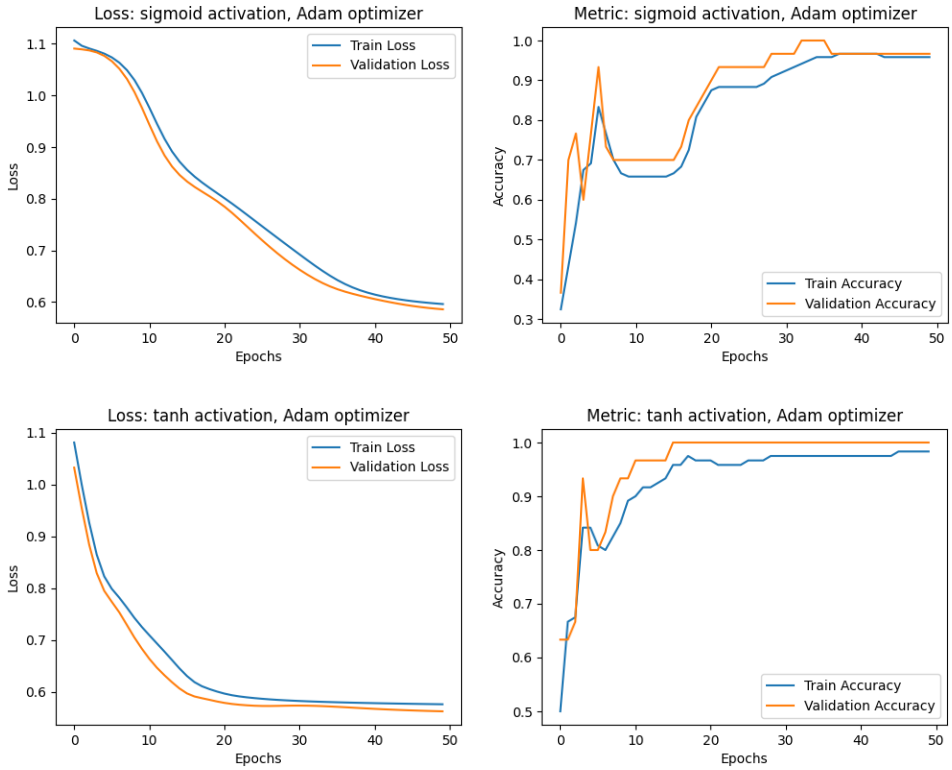
Graphs show predictions by the regression PyTorch model using the Adam optimizer while varying the activation function.

For the PyTorch Regression model, the average test losses for the activation functions are listed below.

	ReLu	Sigmoid	Tanh
Avg Test Loss	14.45	10.88	10.85

The difference in loss is insignificant after the model has been fully trained. While there was little effect on the predictive performance, updating the activation function had a noticeable effect on the rate at which the loss and accuracy changed. The ReLu and Tanh activation functions quickly reduce loss and increase accuracy, reaching a nearly optimal state after only 20-30 epochs. This is opposed to the the Sigmoid function which takes nearly 50 epochs to reach a similar state. These effects are visualized on the graphs below.

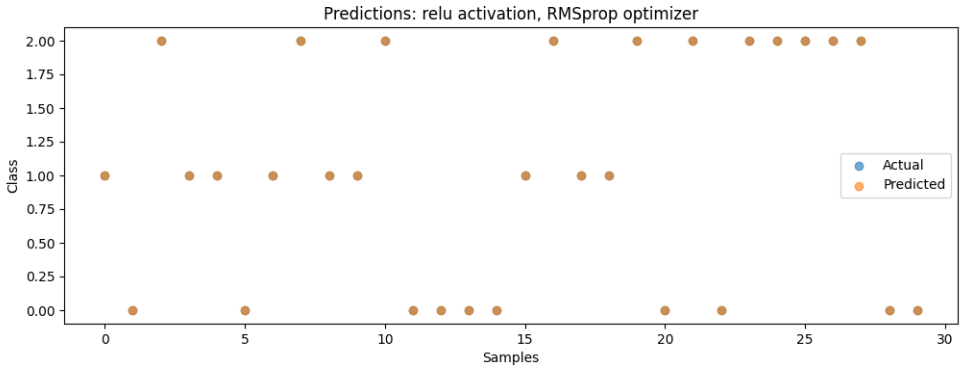


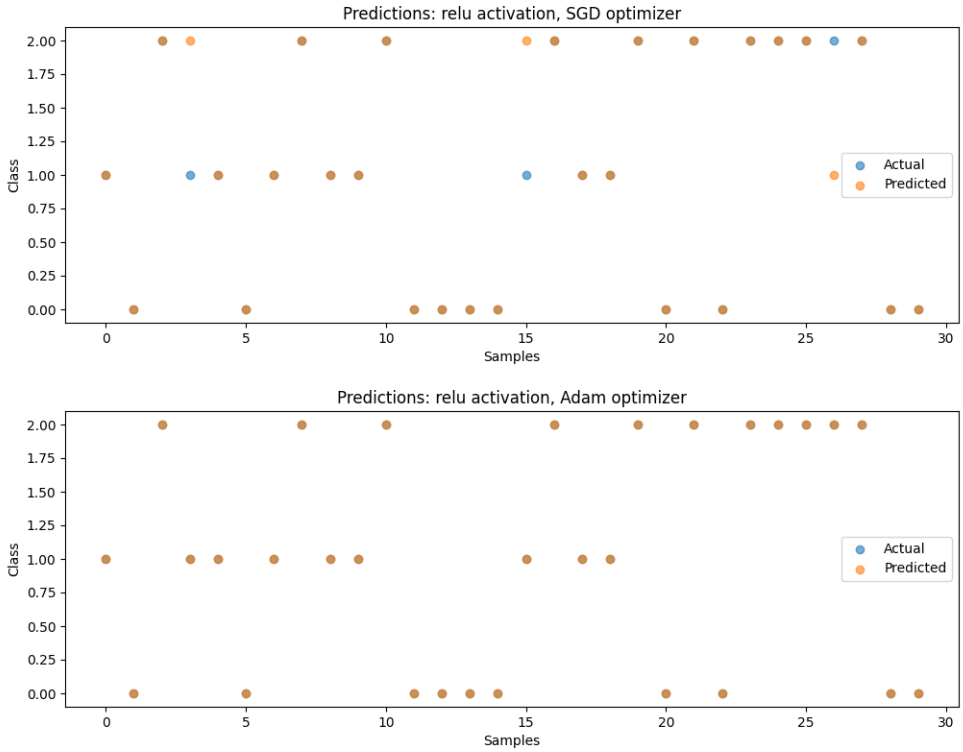


Graphs show the loss and accuracy history of the classification PyTorch model using the Adam optimizer while varying the activation function.

Differences In Optimizers

The choice in optimizer had a greater effect on the performance of the model than the choice of activation function. However, if we exclude SGD, the difference becomes inconsequential. From the graphs below, we can see that the classification predictions made by the Keras model using Adam and RMSprop are both at 100% accuracy, while SGD returns a near 50% accuracy. This behavior is replicated with PyTorch as well.



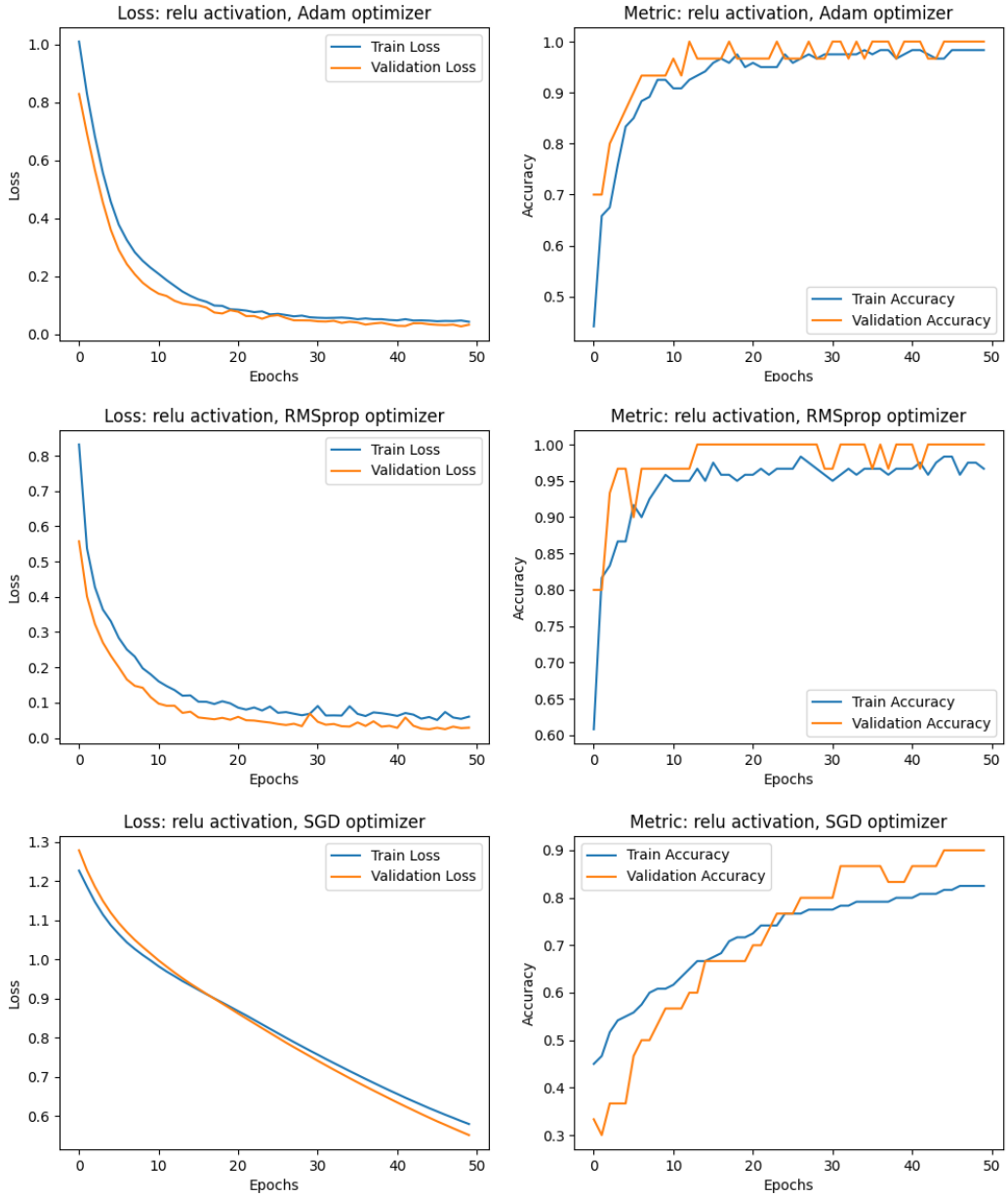


Graphs show the predictions by the classification Keras model using the ReLu activation function while varying the optimizer.

For the PyTorch Classification model, the average test losses and accuracy for the optimizers are listed below.

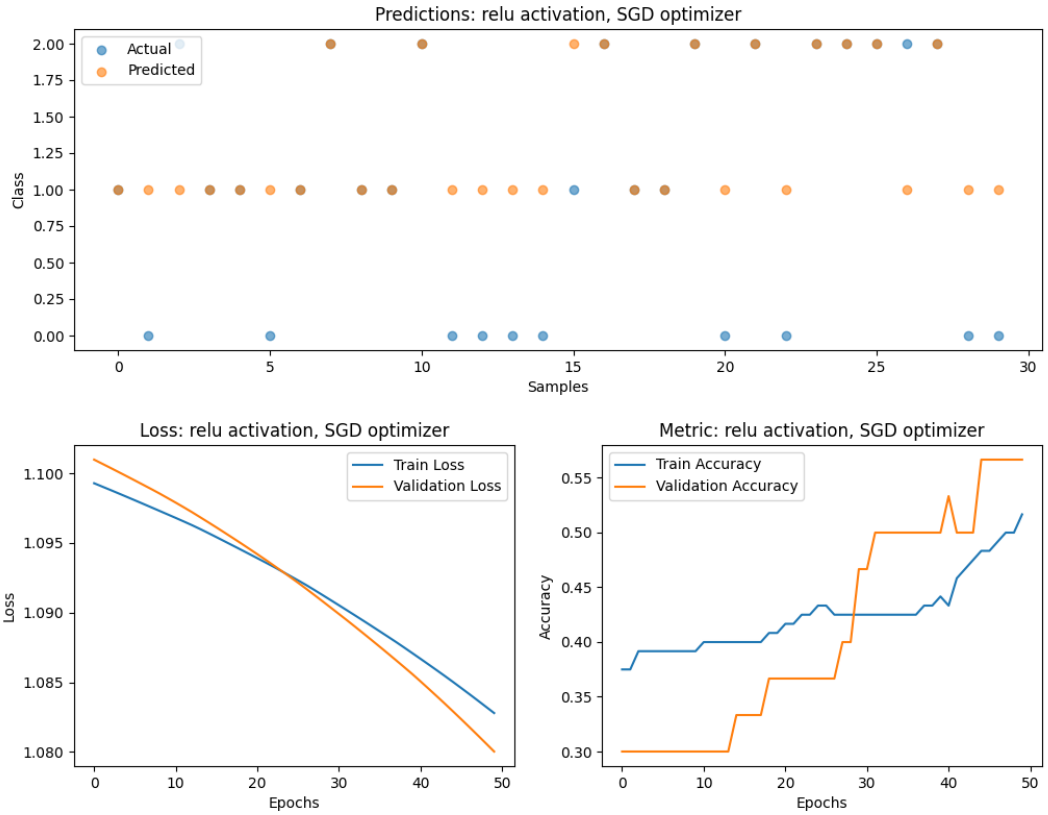
	SGD	Adam	RMSprop
Avg Test Loss	1.08	0.57	0.57
Avg Test Accuracy	0.50	0.99	1.00

*Ex*cluding SGD, the optimizer had little effect on the performance of the classification models. The average Test Loss and Accuracy of Adam and RMSprop are almost identical. From the graphs below, we can see that the rate at which the loss and accuracy are updated is roughly the same.



Graphs show the loss and accuracy history of the classification Keras model using the ReLu activation function while varying the optimizer.

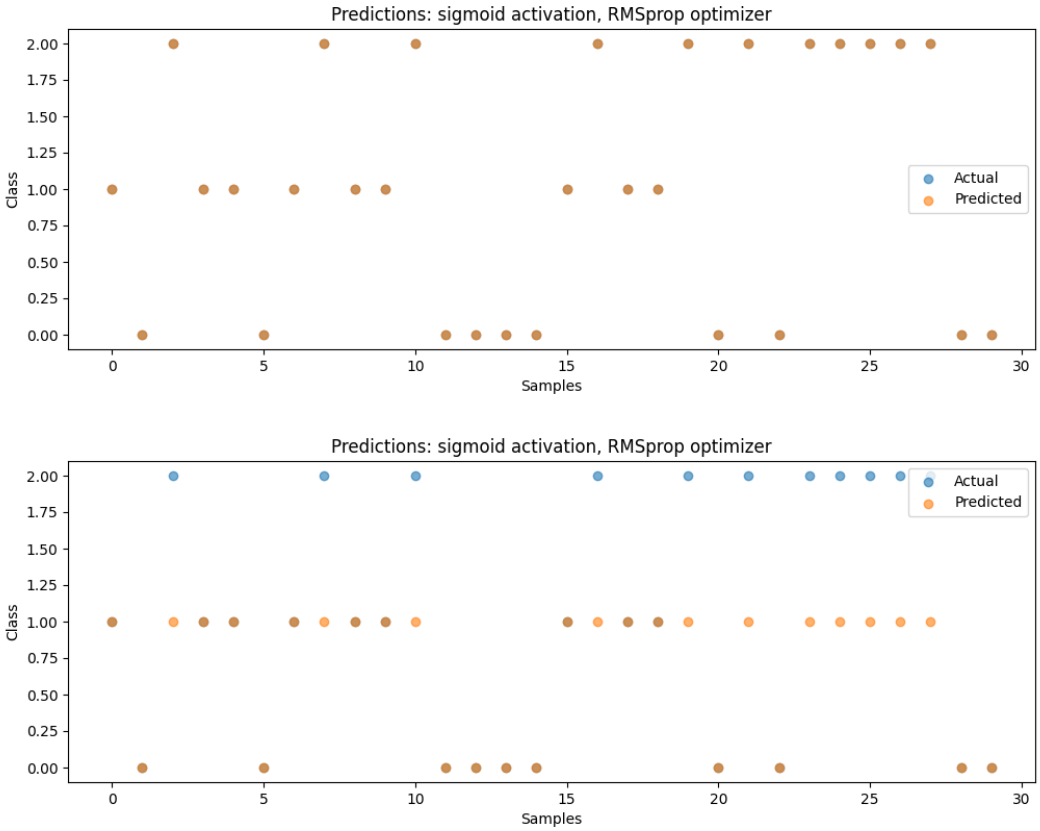
Clearly, Stochastic Gradient Descent fails as an optimizer in the current implementation of both the Keras and PyTorch models. Because SGD updates the model's weights after every data point, the weights grow incredibly fast. With suboptimal hyperparameters like the learning rate and the number of epochs, the model can easily overfit or have poor convergence as evidenced by the high loss and low accuracy.



Graphs show the predictions, loss, and accuracy history of the classification PyTorch model using ReLu and SGD.

Increased Neurons

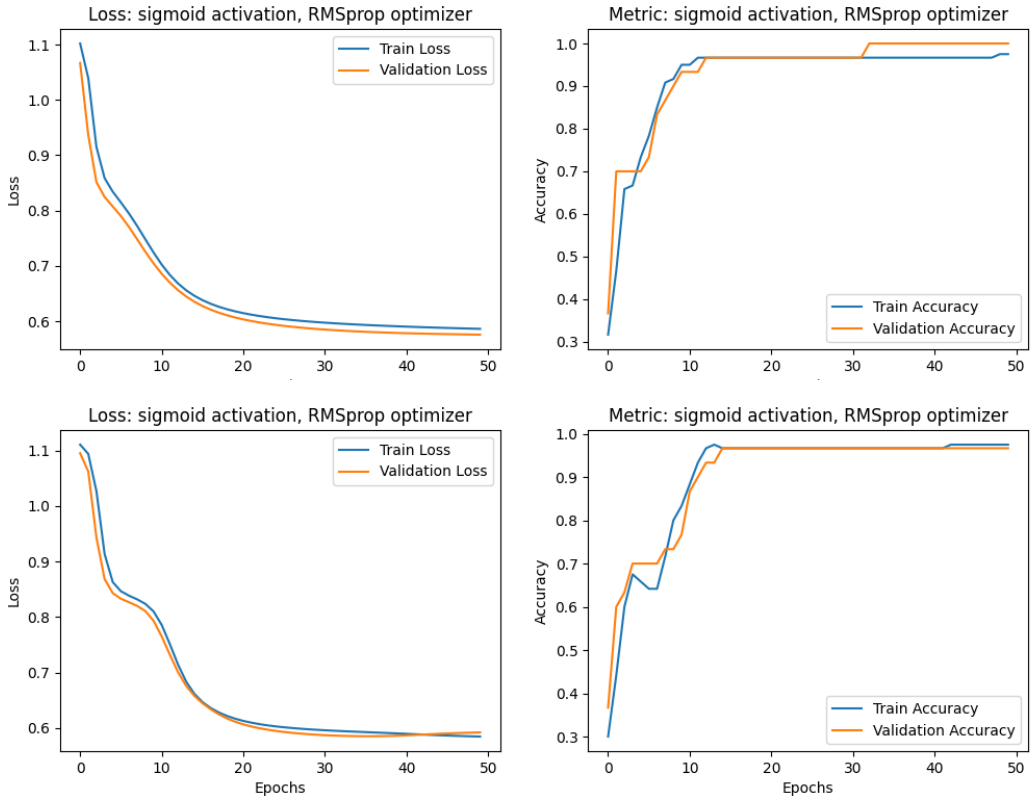
Increasing the number of neurons from to 128 led to a quicker convergence for the Keras models, however, it led to overfitting for the PyTorch models, especially for classification. The jump from 10 to 128 neurons for the classification models created a model far too complex for the data at hand. The PyTorch regression models experienced less overfitting and saw decreased loss compared to Net1 in some cases. Further, the training time of both the Keras and PyTorch models were noticeably increased.



Predictions of the classification PyTorch model for Sigmoid and RMSprop. Top graph is the Net1 predictions and Bottom graph is the Net2 predictions.

Additional Hidden Layers

Adding an additional layer had a less drastic effect on the performance compared to adding more neurons. In most cases, the additional layer allows the model to find more abstract patterns which improves the rate at the loss decreases. The graphs below show how the Net3 model reaches its max accuracy before 15 epochs, while the Net1 Model reaches its max accuracy after 30. However, after the models are fully trained, the difference in accuracy is minor.



Test loss and accuracy of the classification PyTorch model for Sigmoid and RMSprop. Top graphs are from Net1 and Bottom graphs are from Net3.